Antti Aarnio

Web-teknologioiden käyttö junien tietoliikennejärjestelmissä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

29.11.2013

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Antti Aarnio Web-teknologioiden käyttö junien tietoliikennejärjestelmissä 33 sivua 29.11.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Juha-Pekka Kämäri Projektipäällikkö Mikko Rislakki (EKE-Elektroniikka Oy)

Insinöörityön aiheena oli tutkia web-teknologioiden mahdollisuuksia junien tietoliikennejärjestelmissä. Siinä pyrittiin selvittämään, miten web-teknologioita voidaan hyödyntää kuljettajan ja muun junan henkilökunnan työn helpottamisessa sekä matkustajien matkakokemuksen parantamisessa. Erityisen tärkeässä asemassa oli, miten web-teknologiat soveltuvat käytettäväksi EKE-Elektroniikan laitteistoympäristössä.

Työn teoriaosuus käsittelee yleisesti junaympäristön vaatimuksia, ja mitä haasteita nämä vaatimukset aiheuttavat tietoliikennejärjestelmän ja siihen liittyvien sovelluksien toteut-tamiseen. Lisäksi kerrotaan työssä käytetystä EKE-Trainnet-laitteistoalustasta ja siihen liittyvistä moduuleista. Lisäksi tutkittiin web-teknologioiden etuja ja haasteita sekä sitä, miten web-teknologioiden integroiminen EKE-Trainnet-alustaan onnistuu.

Ohjelmointiosuudessa toteutettiin prototyyppiversio HTML5-pohjaisesta käyttöliittymästä, jolla voidaan esittää junan Ethernet-verkosta saatavaa dataa. Käyttöliittymän ohjelmoinnissa käytettiin HTML-, JavaScript- ja CSS-kieliä. Sovellus tarjoaa junan matkustajalle web-sivun, josta matkustaja näkee erilaisia junaan ja matkaan liittyviä tietoja.

Työn tuloksena onnistuttiin integroimaan web-pohjainen käyttöliittymä EKE-Trainnet-alustaan ja sovelluksen kehittämistä päätettiin jatkaa. Tulevaisuudessa on tarkoituksena, että web-pohjaisesta ratkaisusta tehdään osa yrityksen tuotevalikoimaa.

Avainsanat	Junaympäristö, web-teknologiat, HTML5, käyttöliittymät

Author(s) Title Number of Pages Date	Antti Aarnio Use of web technologies in train telecommunication systems 33 pages 29 November 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Lecturer Juha-Pekka Kämäri Project Manager Mikko Rislakki (EKE-Electronics Ltd)

The purpose of the thesis was to explore the possibilities of web-based technologies in train environment. The thesis analyzes how web technologies can be used to enhance the travel experience of travelers and to make the work of train personnel easier. The integration to EKE-Electronics' hardware platform was also one of the key aspects of the study.

The theory chapters of the study explain the requirements of train environment, and what challenges it poses to developing software and hardware. The EKE-Trainnet platform and modules are also introduced and the advantages and shortcomings of web technologies studied.

The prototype produced during this project is a HTML5-based interface, which is used to display data acquired from the train Ethernet network. The programming methods that were used are HTML, CSS and JavaScript. In the HTML5 interface a passenger can view information about the train and the journey.

The integration of web-based interface to EKE-Trainnet platform was successful and it was decided that the development of this solution will continue. In the future it will be a part of the EKE-Trainnet product family.

Keywords	Train environment, HTML5, user interface, web technologies

Sisällys

Lyhenteet

BitTorrent	Tiedonsiirtoon tarkoitettu peer to peer-pohjainen prototolla.
CPS	*Central Processing unit with Serial links module.* EKE-Trainnet-tuoteperheen yksi prosessorimoduleista, joka sisältää sarjalinkkiyhteyden.
CSS	*Cascading Style Sheets.* HTML-sivujen ulkoasun määrittelyyn käytettävä tyyliohje.
DIO	*Digital Input/Output.* EKE-Trainnet-tuoteperheen digitaalinen I/O-moduuli.
ECN	*Ethernet Train Consist.* Junan kiinteiden perusyksiköiden sisäinen Ethernet-verkko.
ED	*End Device.* Junaverkoon liitetty laite. Esimerkiksi junan ovet tai infotaulu.
EN 50155	EN50155 on kansainvälinen rautateillä käytettävien elektronisten laitteiden standardi.
ERU	*Ethernet Routing Unit.* EKE-Trainnet-tuoteperheen Ethernet reititin.
ETB	*Ethernet Train Backbone.* Junan runkoverkko.
ETBN	*Ethernet Train Backbone node.* Junan runkoverkossa oleva laite. Esimerkiksi ERU-moduuli.
FDB	*Function Block Diagram.* PLC-ohjelmoinnissa käytettävä funktio, jonka avulla määritetään sisään- ja ulostulosignaalien välinen logiikka.
GitHub	Verkossa oleva palvelu Git-versionhallintaohjelmiston käyttämiseen.
GSM-R	Rautatiekäyttöön tarkoitettu matkapuhelintaajuusalue.
HTML5	HTML-kielen uusin versio.

IEC 61375 Rautatieliikenteen elektronisten järjestelmien standardi. Määrittelee esimerkiksi TCN-verkon rakenteen.

ISaGRAF PLC-ohjelmointiin tarkoitettu kehitysympäristö.

JavaScript Pääasiassa web-ympäristössä käytettävä ohjelmointikieli.

Lighttpd Avoimen lähdekoodin web-serveri.

LLVM Alunperin *Low Level Virtual Machine*. Kääntäjien kehitykseen tarkoitettu sovelluskehys.

MVB *Multifunction Vechile Bus.* Erityisesti vaunujen sisäisessä tiedonsiirrossa käytetty protokolla.

PIU *Power Input Unit.* EKE-Trainnet-tuoteperheen moduuli, joka ottaa vastaan akkujännitteen ja välittää sen PSV-moduulille.

PLC *Programmable Logic Controller.* Mikroprosessori, jota käytetään automaatioprosessien ohjaamisessa.

PowerPC RISC-arkkitehtuuriin perustuva prosessori.

PSV *Power Supply Unit*. EKE-Trainnet-tuoteperheen virtalähdemoduuli.

RS-485 Sarjaliikenneväylä, johon voidaan liittää useita laitteita samanaikaisesti.

SHA1-tiiviste

Secure Hash Algorithm (versionumero 1). Tiivistealgoritmi, jota käytetään yleisesti tarkistussummien luomiseen. Tarkistussummilla voidaan todeta tiedon eheys.

SparcV9 RISC-arkkitehtuuriin perustuva prosessori.

TCN *Train Communication Network.* Junissa käytetty, IEC 61375 –standardiin perustuva, tietoliikenneverkko.

TCN-Gateway

EKE-Trainnet-tuoteperheen perusyksiköitä, jotka koostuvat vähimmillään prosessorimoduulista, kommunikaatiorajapinnan toteuttavasta moduulista ja virtalähteestä.

Twitter Bootstrap

Twitterin kehittämä ohjelmistokehys, jota käytetään HTML5 käyttöliittymien tekemisessä. Sisältää CSS- ja JavaScipt-kirjastot.

WTB *Wire Train Bus.* Vaunujen välisessä tiedonsiirrossa käytetty protokolla.

x86 Intelin kehittämä prosessoriarkkitehtuuri.

1 Johdanto

Junaliikenteen sujuvuuden sekä matkustajien mukavuuden kannalta on erityisen tärkeää, että junan tietoliikenne toimii luotettavasti ja nopeasti. Yksinkertaistettuna tämä tarkoittaa, että junan henkilökunnalla ja matkustajilla on saatavilla ajantasaista tietoa junan eri järjestelmiltä.

Henkilökunnan tulee olla jatkuvasti tietoinen junan tilasta, jotta matkustajien turvallisuus voidaan taata. Esimerkiksi, jos junan jarrut vikaantuvat, on junan kuljettajan saatava tästä välittömästi tieto. Myös huoltohenkilökunta täytyy pitää tietoisena junassa ilmenneistä vioista, jotta niihin voidaan kohdistaa tarvittavat huoltotoimenpiteet.

EU-lainsäädäntö määrää, että junapalvelun tarjoajan täytyy saattaa matkustajien tietoon vähintään seuraavat asiat: [1.]

- tieto junansisäisistä palveluista
- seuraava asema
- tieto mahdollisista myöhästymisistä
- liikenneyhteydet asemilta
- turvallisuusasiat.

Kuitenkin matkustajamukavuuden lisäämiseksi junapalvelun tarjoajat haluavat pakollisten tietojen lisäksi tarjota paljon muutakin kiinnostavaa ja ajankohtaista tietoa.

Tämän työn tarkoituksena on tutkia, minkälaisia lisäpalveluita, pakollisen tiedon lisäksi, web-teknologioiden avulla voidaan tuottaa sekä matkustajille että junan henkilökunnalle. Lisäksi pohditaan, mitä etuja saavutetaan web-pohjaisella tuotekehityksellä.

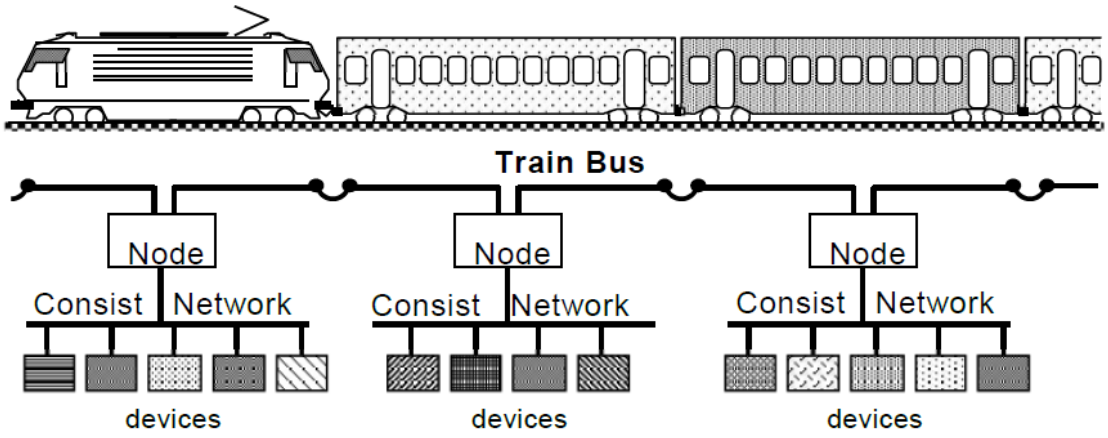
Työn perustana käytetään EKE-Elektroniikka Oy:n (johon tästä eteenpäin viitataan nimellä EKE) junaverkkoarkkitehtuuria. Työ on tilattu saman yrityksen toimesta.

Tämän lisäksi tuotetaan prototyyppiversio web-sivustosta, jonka päätarkoituksena on selvittää, voidaanko HTML5-pohjaista käyttöliittymää voidaan käyttää EKE-Trainnet-verkosta saatavan tiedon esittämiseen. Tutkimuksen pohjalta tehdään päätös siitä kehitetäänkö tuotetta pidemmälle.

2 Laitteistoalusta

Jotta junan alijärjestelmistä saatava tieto voidaan saattaa loppukäyttäjille, täytyy junassa olla junan sisäinen tietoliikenneverkko. Tähän tarkoitukseen EKE on kehittänyt EKE-Trainnet–tuotteen, jonka tarkoitus on yhdistää kaikki junasta saatava tieto yhteen tietoliikenneverkkoon, josta se on helposti saatavilla. EKE-Trainnet on yleisnimitys kaikille EKE:n laitteista koostuville kokonaisuuksille.

Aiemmin perinteinen TCN (Train Communication Network) -verkko rakentui MVB(Multifunction Vechile Bus) -ja WTB(Wire Train Bus) -teknologioiden ympärille, mutta niiden käyttämän RS-485-sarjaliikenneväylän tiedonsiirtonopeus on vain 1 Mbit/s. Näin ollen se soveltuu hyvin pienien tietomäärien miltei reaali-aikaiseen lähettämiseen.



Kuva 1. IEC 61375 -standardin mukainen WTB-pohjainen TCN-verkko [2, s. 41]

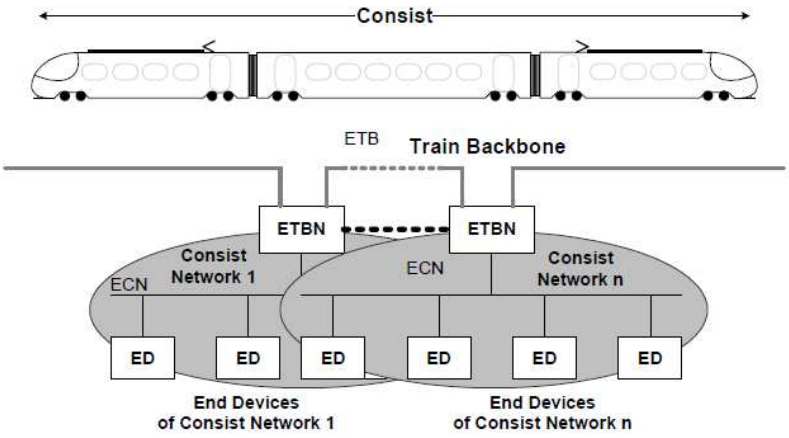
Train Bus eli junan runkoverkko on toteutettu WTB-protokollalla ja junan vaunujen sisäinen consist network on toteutettu yleisesti MVB-protokollan avulla. Muitakin protokollia voidaan kuitenkin käyttää vaunujen sisäisen verkon toteuttamiseen.

Kuitenkin nykyaikana vaatimuksissa on usein myös videomuotoisen datan välittäminen junan sisällä, esimerkiksi turvakamerat. Tämän vuoksi TCN:ää määrittelevää IEC 61375 -standardia on aloitettu laajentamaan siten, että se sisältää myös Ethernetiin perustuvan TCN-verkon. Näin ollen voidaan päästä 100 Mbit/s siirtonopeuksiin, jonka avulla videomuotoisen datan siirtäminen ei ole enää ongelmana. Erityisesti tämä tarkoittaa sitä, että jo vuosia muussa maailmassa käytössä ollut teknologia alkaa siirtyä myös junamaailmaan ja näin ollen avaa oven web-pohjaisille ohjelmistoille.

IEC 61375 standardin mukainen, Ethernet-pohjainen, TCN-verkko koostuu neljästä osasta

- ECN
- ETB
- ETBN
- ED.

ECN on junan kiinteän perusyksikön (consist) sisäinen verkko, joka liittää niiden sisäiset laitteet yhteen (ED). ETB on näiden perusyksiköiden välillä kulkeva verkko, joka liittää ETBN(Ethernet Backbone Node) -laitteet yhteen. Junan kiinteällä perusyksiköllä voidaan tarkoittaa mitä tahansa yhden tai useamman vaunun kokonaisuutta, joita ei eroteta toisistaan normaalikäytössä[3, s. 12.].



Kuva 2. IEC61375-standardin mukainen Ethernet –pohjainen TCN-verkko [4, s. 34.]

Standardissa määritellään verkon topologian lisäksi tarkasti kaikki verkkoon liittyvät ominaisuudet.

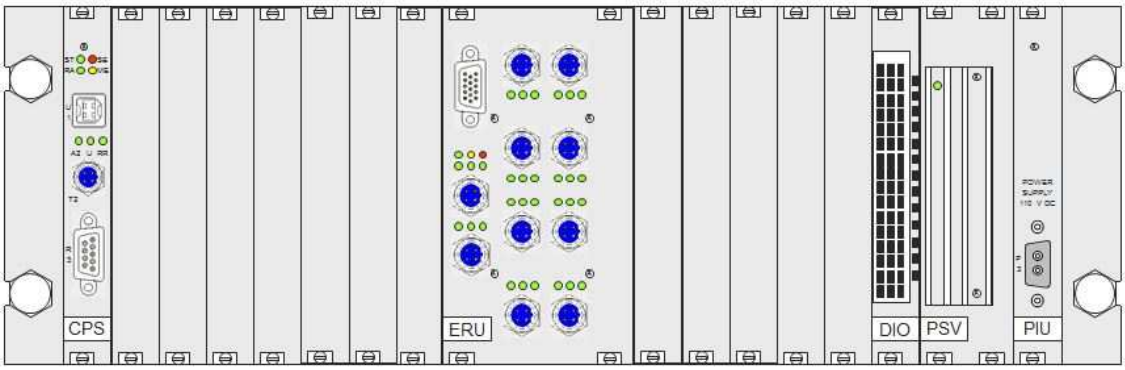
2.1 Laitteisto

Tätä tutkimusprojektia varten luotu verkko koostuu kolmesta TCN Gateway -yksiköstä. TCN Gateway:t ovat EKE Trainnet -tietoliikenneverkon peruskokonaisuuksia, jotka koostuvat vähimmillään keskusyksiköstä, jostakin kommunikaatiorajapinnasta ja virtalähteestä. Normaalisti junassa on yksi TCN Gateway jokaista vaunua kohden.

Tässä projektissa TCN Gateway koostuu seuraavista moduuleista. Moduuleista kerrotaan tarkemmin myöhemmin tässä kappaleessa.

- CPS-moduuli
- ERU-moduuli
- PSV- ja PIU-moduuli
- DIO-moduuli.

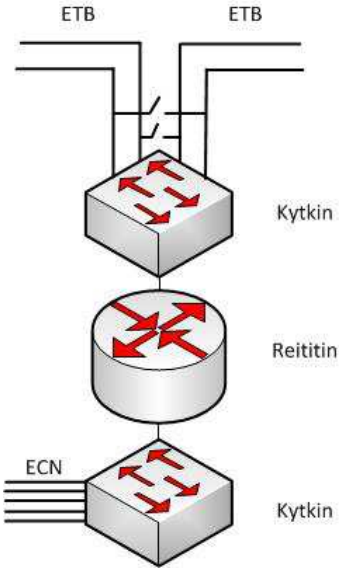
Lisäksi yhden TCN Gateway:n ERU-moduuliin on yhdistetty langattoman yhteyden mahdollistava WLAN-yhteyspiste.



Kuva 3. TCN Gateway.

CPS-moduuli (Central Processing Unit with Serial Links Module) on TCN Gateway:n keskusyksikkö, joka perustuu PowerPC-arkkitehtuuriin. Siinä on Linux-käyttöjärjestelmä, 400 MHz:in suoritin ja 64 MB keskusmuistia.

ERU-moduli (Ethernet Routing Unit) on Ethernet-reititin, joka hallinnoi tietoliikennettä vaunun sisäisen verkon ja vaunujen välisen verkon välillä. Moduulissa on 5 porttia vaunun sisäiseen liikenteeseen ja 4 porttia vaunujen väliseen liikenteeseen. Kuvassa neljä tämä rakenne on kuvattu graafisessa muodossa.



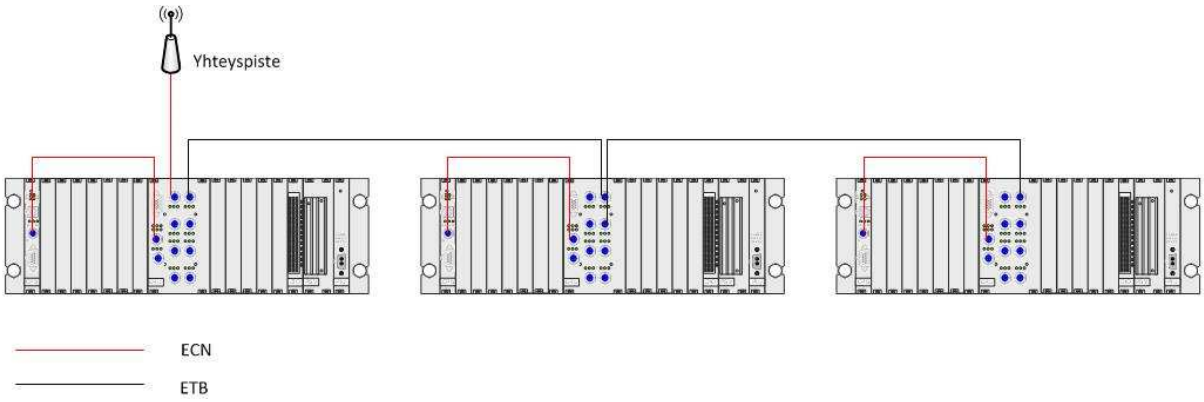
Kuva 4. Osa ERU-moduulin sisäisestä rakenteesta.

PSV- ja PIU-moduleja (Power Supply Unit ja Power Input Unit) käytetään virran saamiseen. PIU-moduuli ohjaa akkujännitteen PSV-moduulin jännitetuloon ja PSV-moduuli ohjaa sen muille moduleille. Tässä projektissa jännitteenä käytetään 110 V tasavirtaa.

DIO-moduuli (Digital Input/Output) on digitaalinen I/O-moduuli, jossa on 24 tulokanavaa.

2.2 Järjestelmäarkkitehtuuri

Tiedonsiirtoväylänä käytetään Ethernetiä ja TCN Gateway:t on yhdistetty toisiinsa ERU-modulien ETB-porttien avulla. ERU-modulin ECN-portteihin on on yhdistetty saman TCN Gatewayn CPS-modulit sekä TCN Gateway 1:ssa WLAN-yhteyspiste.



Kuva 5. Järjestelmäarkkitehtuuri.

Eli aiemmin selvitetyn IEC61375-standardin mukaisesti ERU-modulit ovat ETB- verkon ETBN-yksiköitä ja ERU-moduliin liitetyt CPS-modulit sekä WLAN-yhteyspiste ovat ECN-verkon loppulaitteita (ED).

TCN-laitteiden lisäksi järjestelmään on liitetty I/O-simulaattori, jolla voidaan fyysisesti asettaa DIO-kortin portin tilat haluttuihin asentoihin. Simulaattori koostuu 24 vipukytkimestä, jotka on liitetty suljetun liittimen avulla DIO-korttiin.

3 Junaympäristön vaatimukset

Kun luodaan Ethernet-pohjainen tietoliikenneverkko, esimerkiksi yrityksen sisäiseen tiedonjakeluun (Intranet), on toimenpide yksinkertainen. Markkinoilta löytyy useita miltei valmiita ratkaisuja erikokoisten yritysten tarpeisiin, joiden pohjalta voidaan luoda helposti tarvittava verkko. Pohjimmiltaan junan tietoliikenneverkko on täysin samanlainen, mutta esimerkiksi tärinän ja vaihtelevien olosuhteiden takia, niiden suunnittelussa on otettava huomioon huomattavan suuri määrä eri muuttujia. Nämä muuttujat määritellään EN 50155 -standardissa.

3.1 EN 50155

EN50155 on kansainvälinen rautateillä käytettävien elektronisten laitteiden standardi. Jotta laite voidaan todeta standardin mukaiseksi, täytyy sen olla yhteensopiva seuraavien standardissa määritellyiden osa-alueiden kanssa, Niitä ovat

- elektromagneettinen yhteensopivuus
- iskun- ja tärinänkestävyys
- ilmasto. [5.]

Elektromagneettinen yhteensopivuus tarkoittaa sitä, että laite ei saa häiriintyä ympäristöstä tulevasta elektromagneettisesta energiasta, eikä vastaavasti saa lähettää niin paljon elektromagneettista säteilyä, että se voisi häiritä lähistöllä olevia laitteita. [5.]

Iskun- ja tärinänkestävyys on erityisen tärkeässä asemassa, koska rautateillä liikkuvat laitteet ovat koko elinkaarensa ajan jatkuvasti alttiita iskuille ja tärinälle. Mekaanisten ongelmien lisäksi tämä voi aiheuttaa resonointia mikroprosessoreissa. [5.]

Rautateillä käytettävissä elektronisissa laitteissa täytyy ottaa lisäksi huomioon se, kuinka vaihtelevissa oloissa niitä käytetään. Lämpötila, ilmankosteus ja ilmanpuhtaus ovat tyypillisiä häiriöitä aiheuttavia ilmasto-ominaisuuksia. [5.]

3.2 Liikkuva ympäristö

Vaikka junan sisäinen tietoliikenneverkko pysyy muuttumattomana ja on käytännössä pysyvärakenteinen, on junan jatkuva liike otettava huomioon. Lähijunaliikentessä ja muussa lyhyen välimatkan liikenteessä ongelma ei ole niin suuri, mutta kaupunkien välisessä pitkän matkan liikenteessä tämä tuottaa usein suuriakin ongelmia.

Suurin ongelma on kommunikaatioverkon ja junan ulkopuolisten verkkojen välinen yhteys. Yleisimmin yhteys ulkopuolelle toteutetaan junan katolle sijoitettavan antennin avulla, joka osaa dynaamisesti valita, mitä tiedonsiirtotapaa se käyttää. Normaalisti antennien tukemia yhteystapoja ovat

- 2G/3G/4G
- WLAN
- GSM-R.

Antennit toimivat niin, että jos jonkin yhteystavoista ei ole saatavilla niin antenni vaihtaa automattisesti saatavilla olevaan verkkoon prioriteettijärjestyksessä (WLAN, 4G, 3G,2G ja viimeisenä GSM-R).

GSM-R on vain rautatiekäyttöön tarkoitettu matkapuhelintaajuusalue, jonka taajuus on Euroopassa 876,0–880,0 MHz (lähetys tukiasemalle) ja 921,0–925,0 MHz (vastaanotto tukiasemalta). [6.]

Aina ei ole kuitenkaan saatavilla mitään näistä verkoista, mikä tarkoittaa sitä, että reaaliaikaisista yhteyksistä riippuvaisia toimintoja on hyvin vaikea toteuttaa käytännössä. Tämän vuoksi tarvittava data täytyy ensin saada kokonaisuudessaan junan sisäverkkoon. Sen jälkeen se voidaan vasta esittää junan sisäverkossa olevalle loppukäyttäjälle.

Yksi tapa on BitTorrent-tiedonsiirrosta tuttu datan paloittelu, jolla tarkoitetaan sitä, että kaikki ladattava tieto jaetaan pieniin osiin, jotka voidaan ladata erillisinä paketteina. Kuitenkin BitTorrent-protokollalla tapahtuvasta tiedonsiirrosta poiketen, dataa ladattaisiin vain yhdestä lähteestä kerrallaan. Käytännössä se toimisi niin, että jos halutaan päivittää esimerkiksi hieman suurempi tiedosto, esimerkiksi mainosvideo, niin aina verkkoyhteyden ollessa saatavilla ladataan aina pieni osa koko paketista. Kun koko paketti on saatu ladattua, voidaan se näyttää käyttäjille junan sisäisessä verkossa. Pakettien eheys voidaan tarkistaa käyttämällä SHA1-tiivisteitä. [7.]

Toinen lähestymistapa on, että tiedot päivitetään vain junan ollessa asemilla. Tällöin voidaan käyttää nopeaa WLAN-yhteyttä, joka mahdollistaa suuren tiedonsiirtonopeuden ja nopean datan päivityksen. Ongelmana tässä on kuitenkin se, että pitkillä asemaväleillä tietojen päivitysväli voi olla tunteja, jolloin reaaliaikatietoa vaativia sovelluksia ei ole mahdollista toteuttaa.

Näinollen paras ratkaisu olisi ehkä hybridiratkaisu, joka käyttää hyödyksi molempia ylläkuvailtuja tapoja. Tiedot, jotka eivät ole kriittisiä, voidaan ladata asemilla ja vastaavasti hyvin tärkeät tiedot voidaan ladata matkan varrella datan paloittelua hyväksikäyttäen.

4 Web-pohjaisen tuotealustan edut ja haasteet

Projektin alkuvaiheessa suunnitelmissa oli tehdä eri alustoille omat natiivit sovelluksensa. Kuitenkin projektin edetessä alkoi käydä yhä selvemmäksi, että sitoutuminen yhteen laitealustaan ei ole taloudellisesti kannattavaa. Sen sijaan, että tekisi jokaiselle käyttöjärjestelmälle erillisen ohjelmiston, voidaan käyttöliittymä kehittää HTML5-pohjaisesti. Tällöin huolimatta siitä, millä laitteella loppukäyttäjä haluaa toimia, voidaan luoda vain yksi ohjelmisto. Natiiviohjelmilla on tietyillä osa-alueilla etuja web-pohjaisiin ratkaisuihin verrattuna, erityisesti suorituskyvyssä ja turvallisuudessa.

Seuraavissa luvuissa käydään tarkemmin läpi, mitä etuja, tai vastaavasti haasteita, HTML5-pohjaisella sovelluskehityksellä on.

4.1 Laitteistoriippumaton sovelluskehitys

Laitteistoriippumattomuudella tarkoitetaan sovelluskehityksen näkökulmasta sitä, että sama sovellus toimii kaikilla alustoilla eikä erillisiä sovelluksia tarvita. Sen sijaan, että aika menisi useiden samasta ohjelmasta tehtävien versioiden tekemiseen, voidaan aika käyttää ohjelman ominaisuuksien kehittämiseen. Tämän lisäksi uudet laitteet ovat automaattisesti yhteensopivia sovelluksen kanssa. [8.]

HTML5-ohjelma ei kuitenkaan ole automaattisesti laitteistoriippumaton, vaan kehityksessä tulee ottaa huomioon useita seikkoja kuten

- laitekohtaisten API:en välttäminen
- sisällön responsiivisuus
- ”omituisuuksien” hallinta. [8.]

Kaikki kirjoitettu HTML5- tai JavaScript- koodi ei ole välttämättä laitteistoriippumatonta. JavaScriptillä on kirjastoja, jotka ovat laitteistokohtaisia (WinJS tai Chrome API:t), joten jos niitä käyttää niin sovellus ei enää olekaan laitteistoriippumaton. Tämän vuoksi kehitettäessä laitteistoriippumatonta koodia täytyy olla selvillä eri kirjastojen riippuvuuksista. Myös HTML5:ssä on hiukan eroavaisuuksia eri alustojen välillä, joten tämäkin täytyy ottaa huomioon. [8.]

Sisällön responsiivisuudella tarkoitetaan sitä, miten sovellus muuttuu sen mukaan, minkä kokoiselta ruudulta käyttäjä katsoo sovellusta. Tähän paneudutaan tarkemmin myöhemmässä kappaleessa, joka käsittelee tässä projektissa luodun prototyyppisovelluksen responsiivisia ominaisuuksia.

”Omituisuudet” ovat alustakohtaisia eroavaisuuksia, jotka täytyy ottaa huomioon suunnittelussa. HTML5 kehittyy erittäin nopeasti, josta seuraa, että alustat eivät välttämättä päivity samaa vauhtia kuin itse standardi. Esimerkkinä erovaisuudesta eri alustojen välillä on HTML-lomakkeen ”datetime”-kenttä. iOS5:ssä kyseinen kenttä avaa valitsimen, josta päivämäärän voi valita, kun taas Androidilla tai vanhemmissa iOS versioissa aukeaa vain tekstikenttä, johon päivämäärän voi antaa. [8.]

Osan alustakohtaisista eroavaisuuksista voi kiertää käyttämällä työkaluja kuten Modernizr JavaScriptille tai Normalize.css CSS:älle. Kuitenkaan näidenkään työkalujen avulla ei pysty välttämään kaikkia ongelmia, vaan välillä on myös pakko kirjoittaa alustakohtaista koodia, jolloin on pystyttävä tunnistamaan, mikä alusta on kysessä. [8.]

4.2 Sovelluskehityksen helppous ja sovelluksen ylläpito

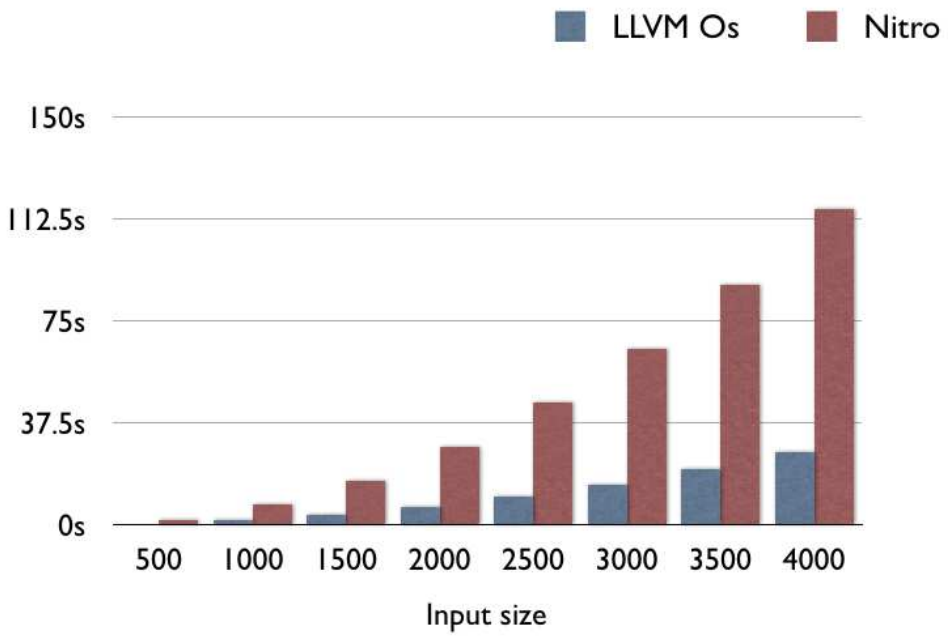
Tällä hetkellä käyttöliittymät EKE-Trainnet-alustalle toteutetaan C/C++-kielillä ja loppukäyttäjän laitteet ovat kiinteitä yksiköitä junan sisällä (käyttöliittymät tehdään C-kielellä) tai kannettavia tietokoneita, joissa ajetaan natiiveja Windows-ohjelmia (C++). Ongelmana on se, että kehitetyt sovellukset ovat erittäin raskaita ylläpitää ja yrityksessä on vain muutama henkilö, ketkä ovat päteviä ylläpitämään ja kehittämään näitä sovelluksia. Jos yksi näistä avainhenkilöistä lähtee yrityksestä tai tulee muuten tarve palkata uusi työntekijä, on korvaavan henkilön löytäminen hyvin aikaavievää ja kallista. Parhaimmassakin tapauksessa uudella työntekijällä kestää kuukausia päästä ymmärrykseen EKE-Trainnet-alustasta.

HTML5-pohjaisella ratkaisulla nämä ongelmat vähenisivät huomattavasti. Erityisesti uuden ja osaavan henkilöstön hankkiminen helpottuisi moninkertaisesti.

4.3 Suorituskyky ja turvallisuus

Suorituskyvyssä ja turvallisuudessa natiivit ohjelmat voittavat HTML5-pohjaiset ohjelmat selvästi. Seuraavasta kuvasta voi nähdä, miten suuri suorituskykyero on LLVM-kääntäjällä käännetyn natiivikoodin ja Applen Nitro JavaScript -enginen välillä.

LLVM on alunperin Ilinoisin yliopistossa kehitetty compiler framwork eli kääntäjien, optimoijien, JIT-koodigeneraattoreiden ja muiden kääntäjiin liittyvien työkalujen kehitykseen tarkoitettu sovelluskehys. LLVM voi luoda koodia x86-, SparcV9- ja PowerPC-alustoille. [9.]



Kuva 6. Suorituskykyvertailu LLVM vastaan Nitro [10.]

Suorituskykyero on noin viisinkertainen eli varsin huomattava. Toinen huomioonotettava seikka on muistinhallinta. Mobiililaitteissa on erittäin rajattu määrä muistia verrattuna työpöytäkoneeseen tai kannettavaan tietokoneeseen, joten kuvien tai muun multimedian käsittelyssä tämä on otettava huomioon. IPhone 4S kaatuu, kun

muistin käyttö ylittää 213MB ja iPad 3 kaatuu 550 MB kohdalla eli käytännössä jo 7 täydellä resoluutiolla otettua kuvaa muistissa kaataa iPhone 4S sovelluksen. [10.]

Turvallisuus on luonnollisesti tärkeä asia kaikissa sovelluksissa ja tällä osa-alueella natiivit ohjelmat ovat selvästi edellä. Web-ohjelmissa on mm. seuraavanlaisia turvallisuusongelmia

- Sivun lähdekoodi on kaikkien nähtävillä.
- Selaimen välimuistiin voi jäädä luottamuksellista tietoa.
- URL:ehin liittyvät haavoittuvuudet. [11.]

Junaympäristössä nämä turvallisuusongelmat korostuvat eritysesti junan henkilökunnalle tarkoitetuissa sovelluksissa, joiden pitää olla ehdottoman turvallisia, jotta asiattomat henkilöt eivät pääse käsiksi junan elintärkeisiin toiminnallisuuksiin. Käytännössä tämä voidaan ratkaista joko luomalla kokonaan fyysisesti erillinen verkko henkilökunnalle ja matkustajille tai virtuaalisesti erotettu verkko.

5 Asiakasvaatimukset

Kun aletaan kehittää sovellusta asiakkaan käyttöön, täytyy selvittää, mitä asiakas haluaa tuotteen tekevän, ja mitä vaatimuksia sen täytyy täyttää. Esitutkimuksessa päädyttiin kolmeen eri käyttäjäryhmään, jolle sovellus voidaan kehittää

- matkustajat
- junan henkilökunta
- varikkohenkilökunta.

Tässä työssä keskitytään matkustajille ja junan henkilökunnalle tarkoitettuun sovellukseen. Seuraavissa kappaleissa selvitetään näiden kahden käyttäjäryhmän tarpeet ja niiden perusteella suunniteltu käyttöliittymä.

5.1 Matkustajainformaatio

Junassa tiedon helppo saatavuus on tärkeässä roolissa. Matkustaja turhautuu hyvin nopeasti, jos hän joutuu etsimään oleellista tietoa liian kauan. Matkustajalla täytyy olla samantien, kun hän avaa käyttöliittymän, saatavilla oleelliset tiedot matkasta eli seuraava asema ja mahdolliset muut viestit, jotka ovat ajankohtaisia junan kulun kannalta. Tärkeimmät matkan aikana matkustajille näytettävät tiedot Accenturen tekemän tutkimuksen mukaan ovat

- muutokset saapumisajassa
- ilmoitukset viivästyksistä tai muista muutoksista
- tiedot mahdollisista liityntäyhteyksistä asemalla. [12.]

Näiden perustietojen lisäksi matkustajalle voidaan luoda lukemattomia eri lisäpalveluita, joiden avulla voidaan matkustajalle luoda koko matkan kestävä saumaton kokemus.

MyTrip-konseptin perustana on yllämainittu ajatus, jossa pyritään siihen, että matkustajalla on kaikki tarvittava tieto käden ulottuvilla siitä hetkestä alkaen, kun hän astuu junaan lähtoasemalla, siihen hetkeen asti, kun hän astuu määränpäässään junasta pois.

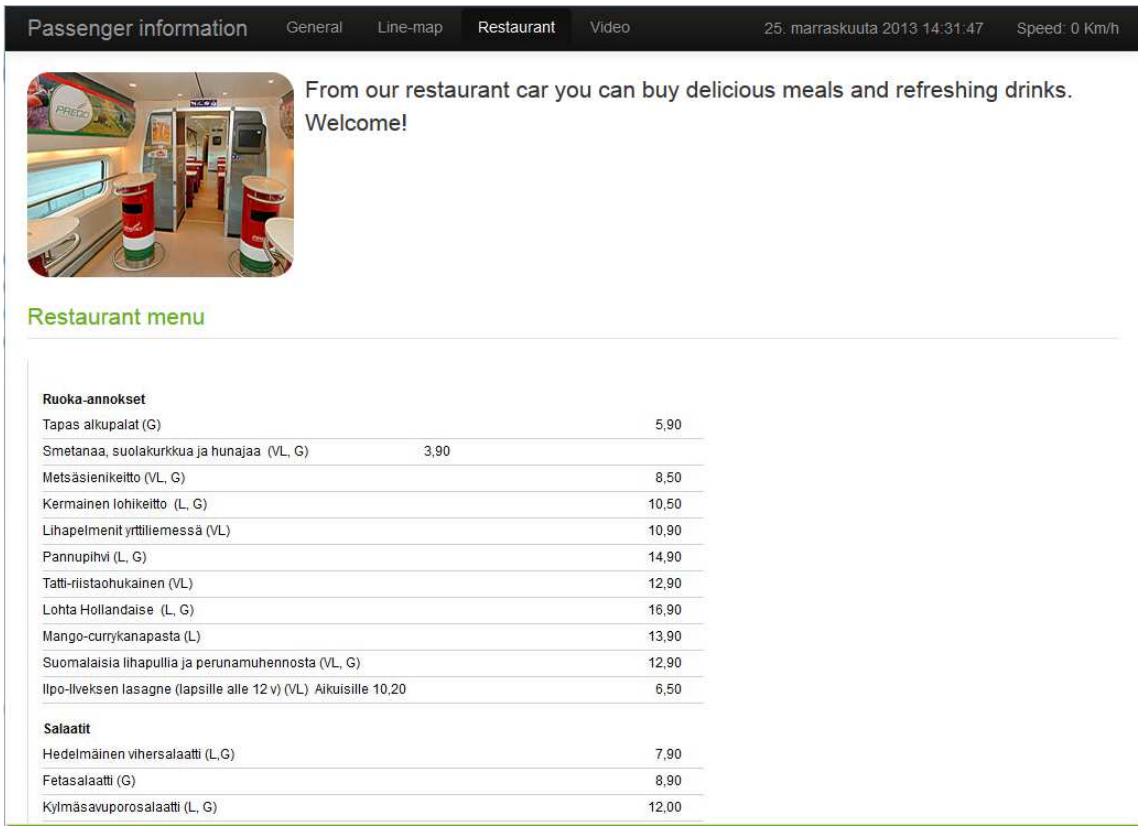
MyTrip-sovelluksen elinkaari alkaa siitä, kun matkustaja saapuu junaan ja yhdistää junan sisäiseen langattomaan verkkoon. Web-sivun etusivulla matkustaja näkee matkansa yleiskuvan, joka sisältää seuraavat elementit

- kellonaika ja päivämäärä
- nykyinen asema ja seuraava asema
- aika jolloin seuraavalle asemalle saavutaan
- junan nopeus

- junan sijainti kartalla.

Sivun yläreunassa on navigointipalkki, josta matkustaja pääsee navigoimaan ohjelman muihin sivuihin.

Pääsivun lisäksi sivusto sisältää mahdollisuuden tutkia junaravintolan ruokalistaa, katsoa videoita (esimerkiksi uutisia) ja sivulta näkee myös koko linja-kartan.



Kuva 7. Esimerkkinäkymä välilehdestä, jossa näkee ravintolan ruokalistan ja navigointirivin.

Edellisessä kuvassa on avattuna välilehti, josta käyttäjä näkee junan ravintolassa saatavilla olevia ruokia. Sivun yläreunassa on näkyvillä navigointipalkki, josta voi vaihtaa välilehteä ja nähdä junan nopeuden sekä päivämäärän ja ajan.

5.2 Junahenkilökunnan informaatio

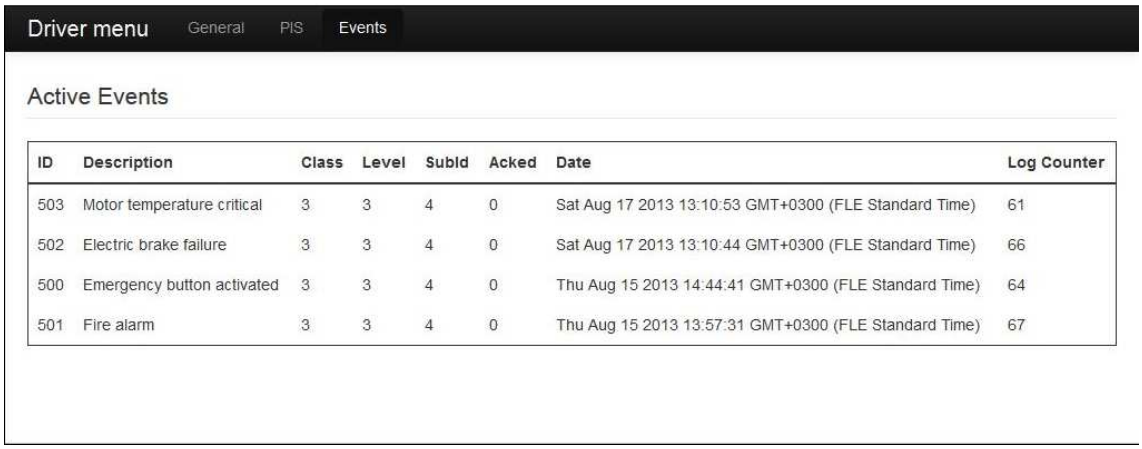
Junan henkilökunnalle tärkeitä asioita ovat reaaliaikainen tieto junan sisäisten järjestelmien tilasta sekä mahdollisuus kommunikoida matkustajien kanssa. Sivu on

ulkonäöllisesti samanlainen kuin matkustajalla, mutta nähtävät näkymät ovat erilaisia sisällöltään.

Etusivulla näkyy

- junan nopeus
- kellonaika ja päivämäärä
- junan kokoonpano.

Junan kokoonpanolla tarkoitetaan järjestystä ja suuntaa, jossa junan vaunut on liitetty toisiinsa. Tämä tieto on saatavilla ERU-modulista. Näiden ominaisuuksien lisäksi kuljettajalla on sivu, josta hän voi lähettää matkustajien käyttöliittymään viestejä, ja sivu, josta kuljettaja näkee sen hetkiset viat junassa. Vikanäkymä esitellään kuvassa 8.



Kuva 8. Näkymä kuljettajan vikalogi-välilehteen

6 Web-teknologioiden integrointi EKE-Trainnet alustaan

Jotta web-sivulle saadaan tietoa EKE-Trainnet-verkosta, täytyy tieto saada sellaiseen muotoon, että se voidaan esittää web-sivulla. Tämä vaatii kolme komponenttia

- ISaGRAF-sovelluksen

- Web-serverin
- HTML5-sovelluksen.

Tässä työssä ISaGRAF:ia käytetään I/O-kortilta tulevien signaalien käsittelyyn ja niiden ohjaamiseen käyttöjärjestelmän Event Logger -rajapinnan käytettäväksi. Tästä rajapinnasta signaalien tiedot voidaan lukea web-serverin käyttöön.

6.1 Kehitysympäristöt

Ohjelmiston kehittämiseen käytettiin kahta työkalua. Ne olivat

- ISaGRAF PLC-kehitystyökalu
- Sublime Text 2-tekstinkäsittelyohjelma.

ISaGRAF-kehitystyökalulla luotiin prosessorimodulin PLC-piiriin ohjelma, jolla saadaan DIO-kortin kanavien tilat näkyviin käyttöjärjestelmässä ja Sublime Text 2-tekstinkäsittelyohjelmaa käytettiin CSS-, HTML- ja Javascript-koodin kirjoittamiseen.

6.2 ISaGRAF PLC-kehitystyökalu

ISaGRAF-kehitystyökalua käytetään EKE:n prosessorimoduleiden logiikkapiirien ohjelmointiin. Tässä projektissa ISaGRAF-kehitystyökalulla luodaan pieni ohjelma, jolla junan vikatiedot voidaan saattaa web-serverin tietoon. ISaGRAF:illa voidaan luoda ohjelmia, joko käyttämällä graafista ohjelmointia tai Structured Text -ohjelmointikieltä. Tässä työssä ei kuvata täydellisesti jokaista askelta, kuinka ISaGRAF-ohjelma tehdään, mutta annetaan kuitenkin selkeä yleiskuva siitä, miten yksinkertaisen projektin luominen ja ohjelmointi tapahtuu.

6.2.1 Uuden projektin konfigurointi

Ensimmäinen askel uuden ISaGRAF-projektin luomisen jälkeen on PLC-määrittelytiedostojen tuominen projektiin. Määrittelytiedostoilla linkitetään projekti EKE:n Linux-käyttöjärjestelmään sekä sen C-funktioihin sekä määritellään datatyypit.

```
[targetUpdate] LINUX_EKE
[end]

[array] log_data256
Type=USINT
Dimension=[1..256]
[end]

[array] below_arr
Type=UDINT
Dimension=[1..2]
[end]

[structure] single_chip_info
[end]
```

Kuva 9. Ote määrittelytiedostosta.

Kun tarvittavat määrittelyt on tuotu, voidaan loput asetukset asettaa oikeiksi.

6.2.2 Ohjelman luominen

Tässä esimerkissa ohjelma toteutetaan ”Function Block Diagram”:illa eli FBD:llä, joka on graafinen tapa ohjelmoida funktioita. Yhden funktion rakenne koostuu aina sisääntulevasta signaalista ja ulosmenevästä signaalista, joiden väliin voidaan määritellä erilaisia loogisia operaatioita ja muita toimenpiteitä.

Kuva 10. ISaGRAF-funktio

Tässä projektissa FBD sisältää viisi funktiota, jotka kaikki ovat rakenteeltaan samanlaisia kuin kuvassa 6 oleva funktio. Funktiot eroavat toisistaan vain yksilöllisen ID-numeron ja sisääntulevan signaalin verran. Funktion sisällä on toinen funktio ”LOG_EVENT”, jota käytetään I/O-signaalin tallentamiseen käyttöjärjestelmän Event Logger -rajapintaan. ”LOG_EVENT”-funktio ottaa vastaan neljä signaalia

- ID

- Act
- Size
- Data.

ID-kenttään annetaan jokaiselle signaalille yksilöllinen tunniste, joka erottaa sen muista tallennettavasta signaaleista. Act-kenttä ottaa vastaan tallennettavan I/O-signaalin, size-kenttään määritellään datan maksimipituus biteissä ja Data-kenttään annetaan DataLogger-tyyppiä oleva struct, johon kaikki signaalit tallennetaan. Nämä signaalit voidaan lukea web-serverille read_events-funktiolla, joka on osa Event Logger -rajapintaa.

Kun ohjelmisto on valmis, se täytyy ladata prosessorille, jonka jälkeen ohjelmassa määriteltyjen signaalien tilat ovat saatavilla käyttöjärjestelmäkomennoilla.

6.3 Sublime Text 2 -tekstinkäsittelyohjelma

HTML5-kehitykseen käytetään Sublime Text 2 -lähdekoodinkäsittelyohjelmaa. Sublime Text 2 on lähdekoodinkäsittelyohjelma, joka osaa eri ohjelmointikielien syntaksin ja lisäksi sisältää automaattisen elementtien täydennyksen. Ohjelman käyttö on hyvin yksinkertaista: ensin valitaan syntaksi, jota halutaan käyttää, ja tämän jälkeen voidaan alkaa kirjoittaa ohjelmaa valittua syntaksia käyttämällä. Tämän lisäksi ohjelmassa on vasemmalla puunäkymä, jossa näkyy kokonaisuudessaan kaikki projektikansiossa olevat tiedostot. Tämä helpottaa navigiontia eri tiedostojen välillä. Ohjelman voi ladata osoitteesta http://www.sublimetext.com/ ja se on saatavilla OS/X:älle, Linuxille ja Windowsille.

6.4 Web-serveri

ERU-modulissa on käytössä lighttpd-serveri, jonka päälle HTML5-sivusto rakennetaan. Lighttpd on avoimen lähdekoodin ohjelma, ja se on suunniteltu erityisesti korkeaa nopeutta vaativiin sovelluksiin [14].

7 HTML5

HTML5 on HTML – kuvauskielen uusin versio, mutta se on myös yleismääritelmä, jolla kuvataan teknologioita, joilla luodaan modernia web-sisältöä. Tärkeimmät teknologiat HTML:än lisäksi ovat CSS ja JavaScript.[14, s. 4.]

Isoimmat uudistukset HTML5:ssä ovat natiivituki videon ja audion toistamiseen suoraan selaimessa sekä canvas-elementti. Canvas-elementtin voidaan toteuttaa JavaScriptin avulla erilaisia toiminnallisuuksia, jotka täytyi aiemmin luoda Adobe Flash -liitännäisen avulla. [14, s. 5.]

Jokainen HTML5-sivu koostuu vähimmillään

- doctype-elementistä
- html-elementistä
- head-elementistä, joka sisältää
    - meta-elementin
    - title-elementti
- body-elementistä.

Doctype-elementissä määritellään tiedoston tyyppi. Tästä selain tietää, minkä tyyppistä tiedostoa se käsittelee. Html-elementti on koko dokumentin juuri-elementti, jonka sisällä kaikki muut elementit sijaitsevat. Head-elementti sisältää tietoa dokumentin sisällöstä (meta-elementin avulla), dokumentin otsikon title-elementillä sekä tämän lisäksi, jos dokumentissa käytetään skriptejä tai css-määrittelyjä, ne ilmaistaan head-elementissä. Body-elementti sisältää dokumentin sisällön. [14, s. 20-21.]

7.1 CSS

CSS eli Cascading Style Sheet on tapa, jolla voidaan määritellä HTML-dokumentin ulkoasu. Tässä luvussa käydään läpi yleisimpiä CSS-tyylien käyttötapoja.

Yksinkertaisin mahdollinen CSS-tyyli on muotoa

```
font-size: 20px;
```

Kuva 11. CSS-tyyli

Kuvassa 7 määritellään font-size-ominaisuudelle arvoksi 20 pikseliä. Jos halutaan määritellä toinen ominaisuus ja arvo, kaksoispisteellä voi erottaa ne toisistaan. Kuitenkaan pelkkä tyylin määritteleminen ei aiheuta mitään muutosta dokumenttiin, vaan tyyli pitää myös liittää johonkin elementtiin. [14, s. 36-38.]

CSS-tyylejä voi liittää elementteihin kolmella eri tavalla

- elementin sisällä
- head-elementissä style-elementin avulla
- ulkoisessa CSS-tyylitiedostossa. [14, s. 38-42.]

Elementin sisällä liittäminen ei ole järkevä vaihtoehto, koska se tekee tyylien päivittämisestä hyvin hankalaa. Style-elementillä tyylin voi liittää seuraavasti

```
<head>
<style type="text/css">
a
{
font-size: 20px;
}
</style>
</head>
```

Kuva 12. Tyylin määrittely style-elementissä

Kuvassa 8 liitetään kaikkiin a-elementteihin ominaisuus font-size, jonka arvoksi asetetaan 20px. Tyyli voidaan myös siirtään ulkoiseen tiedostoon ja nimetä se esimerkiksi ”tyylit.css”. Tämän jälkeen voidaan HTML-dokumentissa link-elementin avulla viitata kyseiseen tiedostoon,jolloin kaikki tiedostossa määritellyt tyylit tulevat automaattisesti dokumenttiin. Link-elementti sijoitetaan head-elementin sisälle.

```
<link rel="stylesheet" href="tyylit.css" type="text/css">
```

Kuva 13. Viittaus ulkoiseen CSS-tyylitiedostoon

Jos mitään yllämainituista tavoista ei ole käytetty, käyttää selain oletustyylejä. Nämä voidaan myös ylikirjoittaa käyttäjän omilla tyyleillä, jolloin jokaiseen web-sivuun, joka selaimella avataan, liitetään tämä tyyli. [14, s.44.]

7.2 JavaScript

JavaScript on ohjelmointikieli, jolla web-sivuista voidaan tehdä interaktiivisempia ja sen avulla voidaan muokata web-sivun sisältöä. JavaScript-skriptejä voidaan määritellä kahdella eri tavalla: HTML-dokumentin sisällä tai erillisessä JavaScript-tiedostossa. Koodiesimerkissä 1 on esimerkkikoodi siitä, miten voidaan nappia painamalla kirjoittaa HTML-dokumenttiin haluttu teksti. Seuraava demo kirjoittaa ruudulle nappia painettaessa: ”Hello JavaScript!”

```
<!DOCTYPE html>
<html>
<body>
<h1>My First JavaScript</h1>
<p id="demo">
</p>
<script>
function myFunction(){
document.getElementById("demo").innerHTML="Hello JavaScript!";
}
</script>
<button type="button" onclick="myFunction()">Click Me!</button>
</body>
</html>
```

Koodiesimerki1. JavaScript- ja HTML-koodit näytölle kirjoittamiseen. [15.]

Esimerkissä määritellään script-elementin sisään JavaScript-funktio ”myFunction”, jota kutsutaan button-elementin onClick-attribuutilla. Div-elementti, johon funktio kirjoittaa, määritellään id-attribuutin avulla, joka määritellään funktion kohteeksi getElementById-metodilla.

Tässä työssä ei käydä sen tarkemmin läpi, miten JavaScript–ohjelmointikieli toimii, mutta myöhemmässä luvussa, jossa esitellään esimerkkisovelluksen JavaScript-toimintoja, käydään läpi miten kyseiset toiminnot on toteutettu.

8 Esimerkkisovellus

Esimerkkisovelluksessa toteutetaan HTML5-käyttöliittymä matkustaja-informaation esittämiseen ja sivu, jolla voidaan selata vikatietoja, jotka saadaan Event Logger -rajapinnasta. Käyttöliittymän totetukseen käytetään HTML-, CSS- ja JavaScript-kieliä. Tämän lisäksi käyttöliittymä käyttää hyväksi web-serverin rajapintoja, joiden avulla päästään käsiksi käyttöjärjestelmältä saataviin tietoihin. Tämä työ ei käsittele

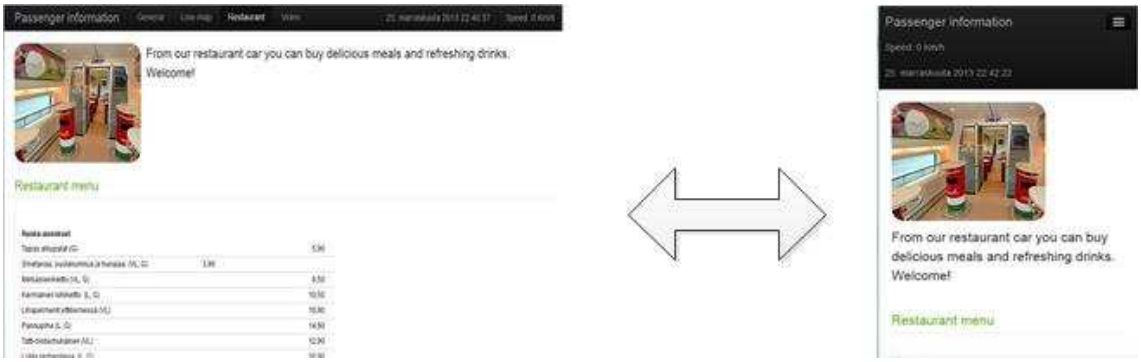
kovinkaan tarkasti sitä, miten sovelluksen taustajärjestelmä toimii, vaan keskittyy HTML5-teknologioiden avulla tuotetun käyttöliittymän luomiseen. Taustajärjestelmän toteutuksesta kerrotaan kuitenkin lyhyesti myöhemmin.

Sovelluksen tarkoituksena on toimia huolimatta siitä, minkälaisella päätelaitteella käyttäjä sitä käyttää eli sivuston täytyy olla responsiivinen. Ulkoasun on tarkoitus olla myös mahdollisimman selkeä ja helppokäyttöinen.

8.1 Responsiivisuus

Perinteinen ei-responsiivinen sivu on suunniteltu siten, että web-sivun rakenne pysyy samana huolimatta siitä, minkälaisella laittella sivustoa katsellaan. Jos käytetään pöytätietokonetta, websivulla näkyy kolme keskikokoista kuvaa mainiosti. Mitä, jos käyttäjä avaakin saman sivun tabletilla? Näkyykö vain yksi kuva ja käyttäjän pitää siirtää selaimen näkymää sivusuunnassa, jotta hän pystyy näkemään kaikki kuvat? Vai loitontaako mobiiliselain kuvat niin pieneksi, että niistä ei saa lainkaan selvää? Responsiivisella web-sivustojen suunnittelulla pyritään ratkaisemaan tämä ongelma.

Responsiivisuudella tarkoitetaan sitä, että sivun sisältö pysyy rakenteeltaan ja ulkonäöltään eheänä, vaikka katseluportin koko muuttuu. Kuvassa 15 on esitetty, kuinka esimerkkiohjelman näkymä muuttuu vaihdettaessa isosta ruudusta pieneen.



Koodiesimerkki 2. Näkymä isolla näytöllä (vasen otos) ja pienellä näytöllä (oikea puoli)

Käytännössä tämä tarkoittaa sitä, että HTML5-koodissa elementeille pitää määritellä erilaiset ominaisuudet sen mukaan, minkä kokoinen katseluportti on.

Responsiivisuuden totetuttamiseen käytetään Twitter Bootstrap -ohjelmistokehystä ja HTML5:en ominaisuuksia.

8.2 Twitter Bootstrap -ohjelmistokehys

HTML5-sivuston pohjana käytetään Twitter Bootstrap -ohjelmistokehystä, joka kehitettiin Twitterin toimesta vuoden 2010 puolivälissä. Sitä kehitettiin yrityksen sisäisesti yli vuosi, kunnes se julkaistiin elokuussa 2011. Tällä hetkellä Bootstrap on GitHubin suosituin projekti. [16.]

Tässä työssä käyttöönotto tapahtuu lataamalla CSS- ja Javascript-kirjastot, jonka jälkeen ne voidaan ottaa käyttöön omassa ohjelmassa. Tämän lisäksi jQuery-kirjasto täytyy ladata ja ottaa käyttöön, jotta Bootstrap toimii.

```
 <script src="js/bootstrap.js"></script>
<link href="css/bootstrap.css" rel="stylesheet">
<link href="css/bootstrap-responsive.css" rel="stylesheet">
```

Koodiesimerkki 3. Bootstrap CSS -kirjastojen käyttöönotto

Navigointipalkkia käytetään siirtymään näkymästä toiseen ja se luodaan käyttämällä Bootstrapin valmiita CSS-tyylejä. Palkki toteutetaan navbar-, navbar-inverse-, navbar-fixed-top- ja navbar-inner-luokkien avulla.

```
<div class="navbar navbar-inverse navbar-fixed-top">
   <div class="navbar-inner">
      <div class="container-fluid">
         <button type="button" class="btn btn-navbar"
            data-toggle="collapse"
            data-target=".nav-collapse">
             <!-- ... -->
         </button>
         <a class="brand">Passenger information</a>
         <div class="nav-collapse collapse">
            <!-- Tähän kirjoitetaan tieto, joka halutaan menevän piiloon-->
         </div>
          <!-- ... -->
```

Koodiesimerkki 4. Navigointipalkin totetuttaminen responsiivisesti

Responsiivisuuden kannalta tärkeä osa koodia on collapse-kohdissa. Ensin painikkeelle annetaan data-toggle-määreen arvoksi ”collapse”, joka määrittää, että käytetään bootstrap.js kirjaston collapse-funktiota. Tämän jälkeen data-target-määreellä osoitetaan valitsin, johon toimenpide halutaan kohdistaa. Tämän jälkeen -elementin sisään kirjoitetaan koko HTML-koodi, mikä halutaan piilottaa. Navbar-fixed-top-luokalla asetetaan navigointipalkki kiinteäksi yläreunaan ja navbar-inverse-luokalla muutetaan palkin väri mustaksi.

Linkit navigointipalkissa on toteutettu välilehtien avulla. Sen sijaan, että jokainen ominaisuus olisi omalla HTML-sivullaan, jokaisella ominaisuudella on oma välilehtensä. Välilehdet saadaan käyttöön data-toggle-määritteen arvolla ”tab” tai ”pill”. Tässä tapauksessa on käytetty arvoa ”pill”.

```
<ul class="nav nav-pills" id="myTab">
                <li class="active">
      <a href="#general" data-toggle="pill">General</a></li>
      <!-- ... -->
</ul>
<!-- ... -->
<div class="tab-pane active" id="general">
<!-- Välilehden sisältö -->
</div>
```

Koodiesimerkki 5. Välilehtien toteus

Ensin määritellään a-elementin href-määreellä välilehden id, joka halutaan avata painamalla linkkiä. Tämän jälkeen voidaan toteuttaa välilehden sisältö div-elementin ”tab-pane”, jonka id on sama kuin aiemmin määritetty, sisään.

8.3 Sivun HTML-toteutus

Käyttöliittymässä on kaksi html-sivua nimeltään passenger.html sekä events.html. Sivut noudattavat HTML5:en mukaista suunnittelua, joten ne sisältävät seuraavat elementit.

```
<html>
   <head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Passenger Information</title>
   </head>
   <body>
   </body>
</html>
```

Koodiesimerkki 6. HTML5-sivun perusrakenne.

Meta-elementeissä määritellään charset-muuttujalla, että sivu käyttää ”utf-8”- merkkejä. Tästä alemmalla rivillä oleva rivi on oleellinen sivun responsiivisuuden kannalta.

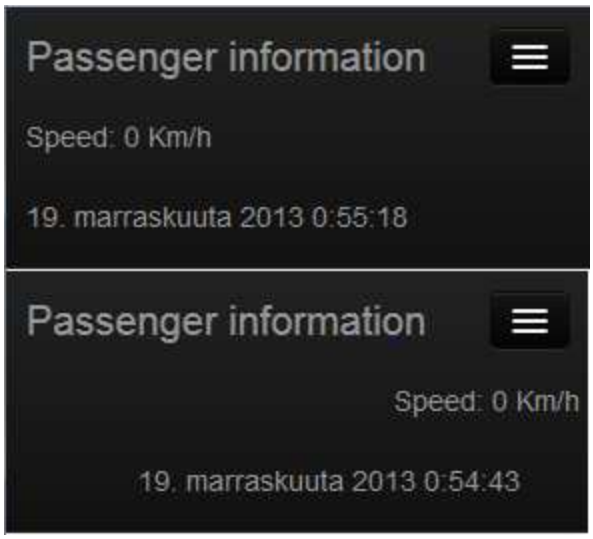
Viewport-määreellä ilmaistaan selaimelle, miten sen tulee piirtää sivu laitteen näytölle. Content-muuttujan arvolla ”width=device-width” asetetaan näkymän leveydeksi laitteen leveys ja ”initial-scale = 1.0” ilmaisee, että kuva täyttää 100 % näkymästä.

Toinen responsiivisuuden kannalta oleellinen kohta HTML-koodissa on media-queryt. Media-queryiden avulla voidaan määritellä, miten sivusto käyttää CSS-tyylejä laitteen ruudun koon muuttuessa. Bootstrap tekee osan media-queryistä automaattisesti, mutta Bootstrapin kirjastossa määriteltyjen media-queryiden lisäksi toteutin yhden media-queryn html-tiedoston sisälle.

```
@media (max-width: 980px){
.navbar-text.pull-right
  {
    float: left;
    padding-left: 5px;
    padding-right: 5px;
  }
}
```

Koodiesimerkki 7. Media-query

Yllä oleva media-query laajentaa Bootstrapin navbar.text-luokkaa pull-right ominaisuudella. Ilman media-querya tekstin asettelu ei toimi oikein ruudun ollessa pienikokoinen, eli alle 980px, vaan se jää paikkaan, joka ei ole visuaalisesti miellyttävän näköinen. Oleellinen ominaisuus on float, jonka arvoksi määritellään left. Tällöin tekstit, joille tämä tyyli asetetaan, liikkuvat niin pitkälle vasemmalle kuin on mahdollista.



Kuva 14. Navigointipalkin tekstit ilman media-querya (alla) ja sen kanssa

8.4 Sivun JavaScript-toteutus

Sivulle on toteutettu kaksi erillistä JavaScript-toiminnallisuutta

- Google Maps (maps.js)
- Ajan ja Päivämäärän näyttäminen (date.js).

Google Maps -toiminnallisuuden toteuttamiseen tarvitaan JavaScript-skripti, Google Maps APIn lataaminen HTML-tiedostossa sekä CSS-tyyli. Malli kartan lisäämiseen HTML-dokumenttiin on otettu Google Developers -sivulta.[12.]

Ensin luodaan HTML-sivulle div-elementti, jolle annetaan id-määreen arvoksi ”map_canvas”. Tämän avulla, kun luodaan JavaScript-osuus, voidaan skriptin sisällä viitata tähän ID:hen. Tämän lisäksi sivulla täytyy ladata Google Maps API sekä lisätä CSS-tyyli map_canvas ID:lle.

```
<script src="https://maps.googleapis.com/maps/api/js?sensor=false"></script>
#map_canvas {
height: 300px;
max-width: 100%;
}
```

Koodiesimerkki 8. Google Maps API:n käyttöönotto ja CSS- tyyli.

Tyylimäärittelyssä määritellään kartalle varattava tila, ja huomattavaa on max-width-ominaisuuden käyttö pelkän width-ominaisuuden sijasta. Jos käyttää width-ominaisuutta niin kartan leveys on vakio, jolloin kartan koko ei muutu vaikka käytettävän laitteen näytön koko muuttuisi. Max-width-ominaisuuden avulla kartan leveys pysyy myös vakiona, mutta suhteessa näytön kokoon eli kartan koko on aina 100 % näytön leveydestä. Näin saadaan jälleen lisättyä sivun responsiivisuutta. Tätä samaa tekniikkaa voidaan käyttää minkä tahansa sivun elementin kanssa. Seuraavaksi tehdään JavaScript-tiedosto, jolla luodaan itse kartta.

Ensimmäisenä täytyy luoda kartta-olio, joka ottaa parametreinään viitteen HTML-dokumentin map_canvas div-elementtiin ja kartan ominaisuudet.

```
var mapCanvas = document.getElementById('map_canvas');
map_options =
 {
center: new google.maps.LatLng(60,10105, 24,56004),
 zoom: 10,
mapTypeId: google.maps.MapTypeId.ROADMAP
}
```

Koodiesimerkki 9. Muuttujat, joissa on viite HTML-dokumentin div-elementtin ja kartan ominaisuudet.

Kartalle on määritelty kolme ominaisuutta

- center
- zoom

- mapTypeId.

Center-ominaisuudella määritellään koordinaatein, mihin kohtaan maailmankartassa Google Maps keskittää näkymän. Tässä tapauksessa koordinaatit on osoitettu Helsingin keskustaan, Kamppiin. Zoom-ominaisuudella määritellään, kuinka lähelle maan tasoa kuvan näkymä on tarkennettu. Suurempi luku tarkoittaa lähempää tarkennusta. MapTypeId-ominaisuus määrittää, minkälainen kartta on. Tässä tapauksessa on valittu tyypiksi tiekartta. [17.]

Kun kuvassa 17 näytetyt muuttujat on luotu, voidaan itse karttaolio luoda.

```
var map = new google.maps.Map(map_canvas, map_options);
```

Koodiesimerkki 10. Kartta-olion luominen.

Tämä ei kuitenkaan vielä riitä, koska funktiota ei vielä suoriteta missään vaiheessa. Suorittaminen tapahtuu lisäämällä kuvan 24 mukaisen rivin funktion ulkopuolelle.

```
google.maps.event.addDomListener(window, 'load', initialize);
```

Koodiesimerkki 11. Funktion suorittaminen

Funktio lisää tapahtumakuuntelijan window-olioon ja kutsuu initialize-funktiota sen jälkeen, kun sivu on latautunut. Jos sivu ladataan ennenkuin sivu on latautunut kokonaan, voi se aiheuttaa ongelmia. Tämä johtuu siitä, että funktion käyttämä div-elementti ei välttämättä ole vielä latautunut. [17.]

Google Mapsin lisäksi JavaScriptiä käytetään ajan ja päivämäärän näyttämiseen. Funktio on varsin yksinkertainen, mutta siinä on kuitenkin mielenkiintoista, miten ajan päivittäminen sekunnin välein on toteutettu.

```
function datetime() {
    var currentDateTime=new Date();
    document.getElementById("datetime").innerHTML=
currentDateTime.toLocaleDateString() + " " +currentDateTime.toLocaleTimeString() ;
    setTimeout(datetime,1000);
```

Koodiesimerkki 12. Date.js-skripti

Ajan päivitys toteutetaan setTimeout-funktiolla, joka toimii siten, että sille annetaan parametriksi ajettava funktio ja odotettava aika millisekunneissa. Ensimmäisen kerran funktio ajetaan HTML-sivun latautuessa window.onload-funktiolla.

```
<script type="text/javascript">
        window.onload = datetime;
    </script>
```

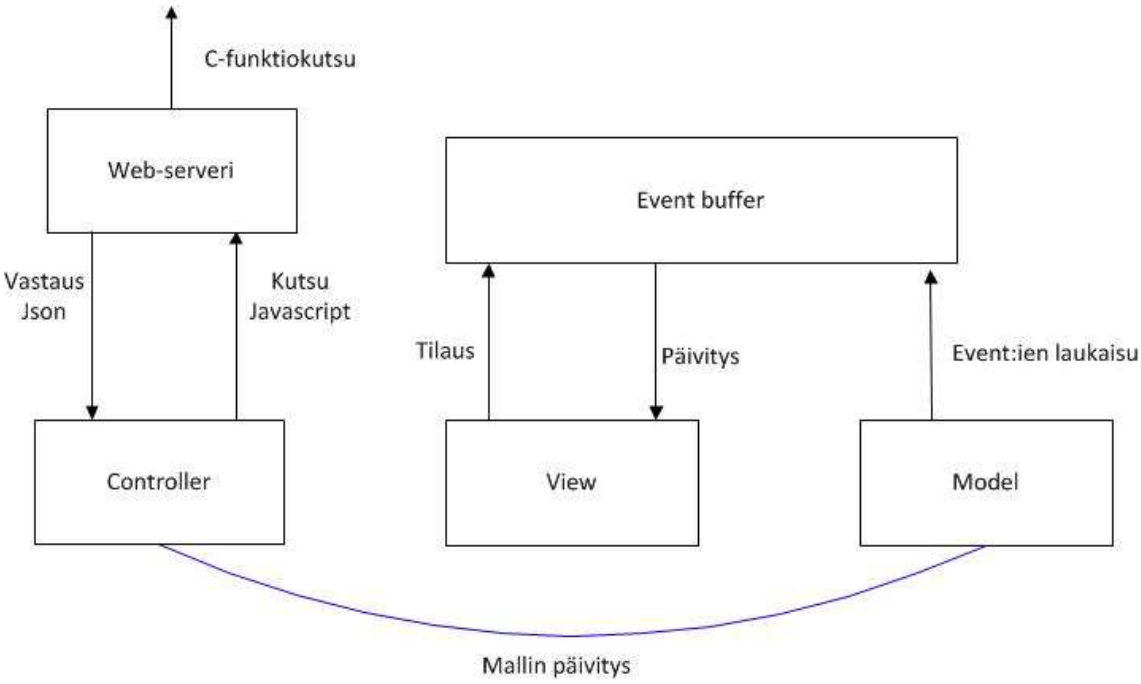
Koodiesimerkki 13. Window.onload

8.5 Tiedon saaminen web-serveriltä

Tässä luvussa kuvataan lyhyesti, miten tiedonsiirto tapahtuu web-serverin ja HTML5-käyttöliittymän välillä.

Sen toteutuksessa on käytetty hyväksi Backbone.js-sovelluskehystä, jonka tarkoituksena on antaa kehittäjälle työkalut hyvin organisoidun web-sovelluksen luomiseen. Backbone ei määrittele tarkasti, miten web-sivu on rakennettava, vaan antaa käyttäjälleen vapauden suunnitella omanlaisensa sivu. [18.]

Alla olevassa kuvassa on esitetty tässä projektissa käytetty sovellusarkkitehtuuri.



Kuva 15. Prototyypin sovellusarkkitehtuuri

HTML-sivulle tiedot saadaan näkyviin hyödyntäen templateja. Esimerkiksi junan nopeus saadaan näytettyä templaten avulla seuraavasti.

```
<script type="text/template" id="speed-template">
   Speed: <span><%= speed %></span> Km/h
</script>
```

Koodiesimerkki 14. Junan nopeuden näyttämiseen käytettävä template

Tällä hetkellä suurin osa web-serveriltä saatavasta tiedosta on simuloitua dataa, ja tiedot luodaan keinotekoisesti web-serverillä. Poikkeuksen tähän tekee aikaisemmin esitelty, ISaGRAF-sovelluksen avulla käytettävä, Event Logger -rajapinta. Tämä käyttää oikeaa DIO-kortilta tulevaa dataa ja sen avulla voi nähdä reaaliaikaisesti DIO-kortin porttien tilan.

9 Yhteenveto

Nykyaikaisissa junissa liikkuvan tiedon määrä kasvaa jatkuvasti ja tämän vuoksi Ethernet-pohjaiset ratkaisut yleistyvät päivä päivältä. Suurimmat ongelmat web-

sovellusten tekemiseen junaympäristöön ei niinkään ole junan sisäinen teknologia vaan se, että junat ovat jatkuvassa liikkessä ja näinollen tiedonsiirron onnistuminen junasta pois ja sisään ei ole koskaan taattua.

Yksi suurimmista eduista perinteiseen junissa nähtävään käyttöliittymien suunnitteluun on se, että web-teknologioihin perustuvat ratkaisut ovat hyvin kevyitä toteuttaa, eikä asiakas ole sitoutunut niin vahvasti järjestelmätoimittajaan. Asiakkaan halutessa voidaan koko käyttöliittymäkehitys jättää asiakkaalle. Kuitenkin vasta tulevaisuudessa, sovelluksen kehittyessä pidemmälle, nähdään, miten esimerkiksi sovelluksen suorituskyky riittää. Työssä tehtyä prototyyppiä on näytetty asiakkaille ja palaute on ollut pääasiassa positiivista ja yleisesti ilmapiiri on se, että junapalveluiden tarjoajat etsivät juuri tämänkaltaisia ratkaisuja. Myös kilpailijoilla on yhä useammin web-pohjaisia käyttöliittymiä heidän tuotteissaan.

Työn pääasiallinen tarkoitus täyttyi onnistuneesti eli työn perusteella pystytään toteamaan, että EKE-Trainnet-alusta soveltuu web-pohjaisten sovellusten ajamiseen. Monia työssä esiteltyjä ominaisuuksia ja ratkaisuja ei ole vielä totetutettu ohjelmaan, mutta työn perusteella tehtiin päätös, että sovelluksen kehittämistä jatketaan ja tulevaisuudessa siitä on tarkoitus tehdä tuote yrityksen valikoimaan.

Lähteet

1 Regulation (EC) No 1371/2007 of the European Parliament and of the Council of 23 October 2007 on rail passengers' rights and obligations. 2007. Verkkodokumentti.<http://eurlex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32007R1371:EN:NOT>. Luettu 28.10.2013.

2 IEC 61375-2-1.Electronic railway equipment - Train communication network (TCN) - Part 2-1: Wire Train Bus (WTB). 2012.Brysseli: European Committee For Electrotechnical Standardization

3 IEC 61375-1. Electronic railway equipment - Train communication (TCN) - Part 1: General architecture. 2012. Brysseli: European Committee For Electrotechnical Standardization.

4 IEC 61375-2-5. Electronic railway equipment - Train communication (TCN) - Part 2-5: Ethernet Train Backbone. 2012. Brysseli: European Committee For Electrotechnical Standardization.

5 EN50155 Type Approval Guide eBook. Verkkodokumentti. EN50155 Made Easy <http://www.en50155.com/overview/>. Luettu 26.10.2013.

6 GSM-R - Global Standard for Mobile Communications Rail. Verkkodokumentti. Willtek Test Instruments. <http://www.willtek.com/english/technologies/gsmr>. Luettu 25.11.2013.

7 Cohen,B. 2008. The BitTorrent Protocol Specification. Verkkodokumentti. Bittorrent.org. <http://bittorrent.org/beps/bep_0003.html>. Päivitetty 20.10.2012. Luettu 28.10. 2013.

8 Building cross-platform apps with HTML5. 2013. Verkkodokumentti. Intel Developer Zone. <http://software.intel.com/en-us/html5/articles/building-cross-platform-apps-with-html5>. Luettu 1.11.2013.

9 Lattner, C & Adve,V. 2004. The LLVM Compiler Framework and Infrastructure Tutorial. 2004. Verkkodokumentti. <http://llvm.org/pubs/2004-09-22-LCPCLLVMTutorial.html>. Luettu 25.11.2013.

10 Crawford, D. 2013. Why mobile web apps are slow. Verkkodokumentti. <http://sealedabstract.com/rants/why-mobile-web-apps-are-slow/>. Luettu 1. 11. 2013.

11 Rende, J. 2013. Native vs. HTML5 – looked at objectively, the debate is over. Verkkodokumentti. Appcelerator.

<http://thinkmobile.appcelerator.com/blog/bid/284174/Native-vs-HTML5-looked-at-objectively-the-debate-is-over>. Luettu 5.11.2013.

12 Putting the passenger in the driver's seat: Accenture consumer survey on Western European rail services. Verkkodokumentti. Accenture. <http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Consumer-Survey-on-Western-European-Rail-Service.pdf>. Luettu 5.11.2013.

13 Lighttpd. Verkkodokumentti. <http://en.wikipedia.org/wiki/Lighttpd>. Luettu 8.11.2013

14 Freeman, A. 2011. The Definitive Guide to HTML5. Apress.

15 W3Schools.com – Try It Your Self. Verkkodokumentti. <http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_event>. Luettu 25.11.2013.

16 Bootstrap (front-end framework). Verkkodokumentti. <http://en.wikipedia.org/wiki/Bootstrap_%28front-end_framework%2>. Luettu 19.11.2013

17 Adding a Google Map to your website. 2013. Verkkodokumentti. Google Developers. <https://developers.google.com/maps/tutorials/fundamentals/adding-a-google-map>. Luettu 19.11.2013

18 Backbone.js. Verkkodokumentti. <http://backbonejs.org/#introduction>. Luettu 20.11.2013.